

CLAIMS:

What is claimed is:

1. A method of testing a device driver on a data processing system, said method comprising:

allocating, by an operating system, a data space for executing a device driver;

executing the device driver as an application on top of the operating system to test the device driver;

monitoring to detect whether a request made by the device driver specifies a target address within the data space; and

in response to detecting the target address for the request being made outside of the data space, trapping on the target address for the request and executing a data exception handler that emulates a target device.

1 2. The method according to Claim 1, further comprises:

2 setting up, by the operating system, the data
3 exception handler.

1 3. The method according to Claim 2, wherein the setting
2 up step further comprises:

3 calling the operating system to install a data
4 exception handler facility containing the data exception
5 handler into a vector of an interrupt table for the
6 operating system; and

7 calling the data exception handler facility to
8 register data that is used to determine when the data
9 exception is to be taken wherein the data include each of
10 various target addresses at which the data exception is
11 to be taken, lengths of the target addresses, and user
12 callback routines for calling back to the application
13 into a data exception database table.

14 4. The method according to Claim 3, further comprises:

15 passing, by the data exception handler facility, the
16 data space to the vector of the interrupt table.

1 5. The method according to Claim 4, wherein the data
2 exception handler is a memory mapped input/output (IO)
3 exception handler comprising a parser with a disassembler
4 that is used for disassembling and identifying an
5 assembler instruction of the request.

1 6. The method according to Claim 5, wherein the
2 assembler instruction is identified as a load command.

1 7. The method according to Claim 5, wherein the
2 assembler instruction is identified as a store command.

1 8. The method according to Claim 3, wherein the data
2 exception handler further comprises:

3 saving, into a memory stack, data for the request
4 which includes at least the target address for the
5 request, content data relating to the request, and a data
6 address range for the content data;

7 determining whether the target address that is saved
8 into the memory stack is within the database exception
9 database table;

10 immediately terminating the method of testing if the
11 target address is not within the database exception
12 database table;

13 disassembling the data for the request into
14 disassembled information and passing the disassembled
15 information into a respective one of the user callback
16 routines stored in the data exception database table
17 wherein the respective one of the user callback routines
18 that is used is based on the request and is for emulating
19 a behavior of the target device;

20 emulating, by the respective one of the user
21 callback routines, the behavior of the target device;

22 setting a next instruction address to an address in
23 the memory stack that is after the currently saved target
24 address; and

25 unwinding the memory stack by the data exception

26 handler to return control back to the application.

1 9. The method according to Claim 8, further comprising:

2 copying, as necessary, the content data by the
3 respective one of the user callback routines from another
4 data space for the respective one of the user callback
5 routines based on the request.

1 10. The method according to Claim 1, wherein the data
2 exception handler is a software emulator of the target
3 device and further comprises:

4 using a hardware emulator to test the application
5 for the device driver if the hardware emulator is coupled
6 to the data processing system; and

7 using the software emulator to test the application
8 for the device driver if the hardware emulator is not
9 coupled to the data processing system.

10 11. The method according to Claim 1, further comprises:

11 determining whether the application for the device
12 driver has finished executing;

13 continuing with the executing of the application if
14 the application has not finished executing; and

15 terminating the method of testing when the
16 application has finished executing.

1 12. A data processing system for testing a device
2 driver, said data processing system comprising:

3 a processor and a memory system, wherein:

4 said processor executes an operating system that
5 allocates a data space for executing applications and
6 executes a device driver as an application on top of the
7 operating system to test the device driver; and

8 said processor and said memory system, responsive to
9 detecting a request by the device driver specifying a
10 target address outside of the data space, trap on the
11 target address and execute a data exception handler that
12 emulates a target device of the device driver.

1 13. The data processing system according to Claim 12,
2 wherein said processor and said memory system call a data
3 exception handler facility to register data that is used
4 to determine when a data exception is to be taken,
5 wherein the data include each of various target addresses
6 at which the data exception is to be taken, lengths of
7 the target addresses, and user callback routines for
8 calling back to the application into a data exception
9 database table.

1 14. The system according to Claim 13, wherein said
2 processor and said memory system call the operating
3 system to install the data exception handler facility
4 containing the data exception handler into a vector of an
5 interrupt table for the operating system.

1 15. The system according to Claim 14, wherein the data
2 exception handler facility passes the data space to the
3 vector of the interrupt table.

1 16. The system according to Claim 15, wherein the data
2 exception handler is a memory mapped input/output (IO)
3 exception handler that is stored in the memory system and
4 comprises a parser with a disassembler that is used for
5 disassembling and identifying an assembler instruction of
6 the request.

1 17. The system according to Claim 16, wherein the
2 assembler instruction is a load command.

1 18. The system according to Claim 16, wherein the
2 assembler instruction is a store command.

1 19. The system according to Claim 14, wherein the data
2 exception handler determines whether the target address

3 is within the data exception database table and,
4 responsive to said determination, immediately terminates
5 testing of the device driver if the target address is not
6 within the data exception database table, and, emulates,
7 by the respective one of the user callback routines, the
8 behavior of the target device if the target address is
9 within the data exception database table.

1 20. The system according to Claim 19, wherein the data
2 exception handler further:

3
4 copies, as necessary, the content data by the
5 respective one of the user callback routines from another
6 data space for the respective one of the user callback
7 routines based on the request.

8 21. The system according to Claim 12, wherein the data
9 exception handler is a software emulator of the target
10 device and wherein:

1 a hardware emulator is used to test the application
2 for the device driver if the hardware emulator is coupled
3 to the data processing system; and

4 the software emulator is used to test the
5 application for the device driver if the hardware
6 emulator is not coupled to the data processing system.

1 22. A program product for testing a device driver on a
2 data processing system comprising:

3 instruction means for allocating, by an operating
4 system, a data space for executing a device driver;

5 instruction means for executing the device driver as
6 an application on top of the operating system to test the
7 device driver;

8 instruction means for monitoring to detect whether a
9 request made by the device driver specifies a target
10 address within the data space;

11 in response to detecting the target address for the
12 request being made outside of the data space, instruction
13 means for trapping on the target address for the request
14 and executing a data exception handler that emulates a
15 target device; and

16 computer usable media bearing said instruction
17 means.

1 23. The program product according to Claim 22, further
2 comprises:

3 instruction means for setting up, by the operating
4 system, the data exception handler.

1 24. The program product according to Claim 23, wherein
2 the instruction means for setting up further comprises:

3 instruction means for calling the operating system
4 to install a data exception handler facility containing
5 the data exception handler into a vector of an interrupt
6 table for the operating system; and

7 instruction means for calling the data exception
8 handler facility to register data that is used to
9 determine when the data exception is to be taken wherein
10 the data include each of various target addresses at
11 which the data exception is to be taken, lengths of the
12 target addresses, and user callback routines for calling
13 back to the application into a data exception database
14 table.

1 25. The program product according to Claim 24, further
2 comprises:

3 instruction means for passing, by the data exception
4 handler facility, the data space to the vector of the
5 interrupt table.
6

1 26. The program product according to Claim 22, wherein
2 the data exception handler further comprises:

3 instruction means for saving, into a memory stack,

4 data for the request which includes at least the target
5 address for the request, content data relating to the
6 request, and a data address range for the content data;

7 instruction means for determining whether the target
8 address that is saved into the memory stack is within the
9 database exception database table;

10 instruction means for immediately terminating the
11 testing of the device driver if the target address is not
12 within the database exception database table;

13 instruction means for disassembling the data for the
14 request into disassembled information and passing the
15 disassembled information into a respective one of the
16 user callback routines stored in the data exception
17 database table wherein the respective one of the user
18 callback routines that is used is based on the request
19 and is for emulating a behavior of the target device;

20 instruction means for emulating, by the respective
21 one of the user callback routines, the behavior of the
22 target device;

23 instruction means for setting a next instruction
24 address to an address in the memory stack that is after
25 the currently saved target address; and

26 instruction means for unwinding the memory stack by
27 the data exception handler to return control back to the
28 application.

1 27. The program product according to Claim 26, further
2 comprising:

3 instruction means for copying, as necessary, the
4 content data by the respective one of the user callback
5 routines from another data space for the respective one
6 of the user callback routines based on the request.

1 28. The program product according to Claim 22, wherein
2 the data exception handler is a software emulator of the
3 target device and further comprises:

4 instruction means for using a hardware emulator to
5 test the application for the device driver if the
6 hardware emulator is coupled to the data processing
7 system; and

8 instruction means for using the software emulator to
9 test the application for the device driver if the
10 hardware emulator is not coupled to the data processing
11 system.

12 29. The program product according to Claim 22, further
13 comprises:

14 instruction means for determining whether the
15 application for the device driver has finished executing;

16 instruction means for continuing with the executing
17 of the application if the application has not finished
18 executing; and

19 instruction means for terminating the testing of the
20 device driver when the application has finished
executing.